Zusammenfassung Info I/II (Grundlagen & Techn. Informatik)

Boole'sche Algebra: · **Axiome:** (i) $A \vee B = B \vee A$, $A \wedge B = B \wedge A$ (ii) $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$, $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ (iii) $A \vee 0 = A$, $A \wedge 1 = A$ (iv) $A \vee \neg A = 1$, $A \wedge \neg A = 0$

$\Rightarrow$ $A \vee 1 = 1$, $A \wedge 0 = 0$, $A \vee A = 1$, $A \wedge A = A$, $A \vee (B \vee C) = (A \vee B) \vee C$, $A \wedge (B \wedge C) = (A \wedge B) \wedge C$, $\overline{A \vee B} = \bar{A} \wedge \bar{B}$, $\overline{A \wedge B} = \bar{A} \vee \bar{B}$, $\bar{\bar{A}} = A$

Schaltalgebra:

$A \cdot 0 = 0$, $A \cdot 1 = A$, $A \cdot \bar{A} = 0$, $A \cdot B = B \cdot A$

$A + 0 = A$, $A + 1 = 1$, $A + \bar{A} = 1$, $A + B = B + A$

$(A \cdot B) \cdot C = A \cdot (B \cdot C)$, $A \cdot A = A$, $A \cdot (A + B) = A$, $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$

$(A + B) + C = A + (B + C)$, $A + A = A$, $A + (A \cdot B) = A$, $A + (B \cdot C) = (A + B) \cdot (A + C)$

$\overline{\bar{A}} = A$

Umkehrsatztheoreme: (i) de Morgan: $\overline{A \cdot B \cdot C \cdot \ldots} = \bar{A} + \bar{B} + \bar{C} + \ldots$, $\overline{A + B + C + \ldots} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \ldots$

(ii) Shannon: $\bar{F}(A, B, C; \bar{A}, \bar{B}, \bar{C}) = F(\bar{A}, \bar{B}, \bar{C}; A, B, C)$ | · wird + wird ·

Darstellungsformen schaltalgebr. Funktionen:

· Bei M Eingangsvariablen: · 2^M Zeilen d. Tabelle

· $2^{(2^M)}$ mögliche Funktionen

Boole'sche Normalform: (i) Separationstheorem: $F(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) = x_i \cdot F(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) + \bar{x}_i \cdot F(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$

$\Rightarrow F(x_1, \ldots, x_n) = \overbrace{x_1 x_2 \cdots x_n}^{\text{Min-Term}} F(1, 1, \ldots, 1) + \bar{x}_1 \cdot x_2 \cdots x_n F(0, 1, \ldots, 1) + \ldots$

· Die „Disjunktive Normalform" (DNF) ist die disjunkte Verknüpfung aller Vollkonjunktionen (Min-Terme), für die der Wert $F(\ldots) = 1$ ist

· Die „konjunktive Normalform" (KNF) ist die konj. Verknüpfung aller Volldisjunktionen (MAX-Terme), für die der Wert $F(\ldots) = 0$ ist

(Negation d. DNF)

Grafische Darstellung: (i) VENN-Diagramm: entspricht Mengenalgebraischer Darstellung

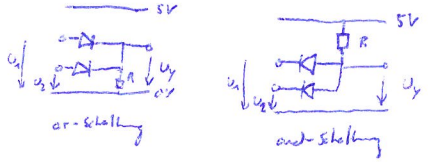
(ii) Karnaugh-Diagramm: enthält 2^M Felder

Inhalt: Wert d. Min-Terms

Vereinfachung: Gebiete mit 1 zusammenfassen (Gebietsgröße: 2^k mit $k = 1, \ldots, j$ bis M)

Logische Grundschaltungen:

Diodenlogik (DL):

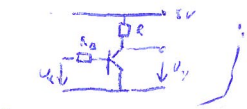


Merkmale: · Logikpegel verändern sich

· keine Regeneration d. Logikpegel

· keine Invertierung möglich

Logikinverter:



Bei Dimensionierung: (i) Ermittlung d. Basisstrom: $I_B = \frac{U_x - U_{BE}}{R_B}$

(ii) Lastgrad: unbelastet: $I_L = \frac{U_B - U_{CE}}{R_C}$

$\hookrightarrow I_C = 0 \Rightarrow U_{CE} = U_B$, $U_{CE} = 0 \Rightarrow I_C = \frac{U_B}{R_C}$ } Grenzen im Ausgangskennlinienfeld

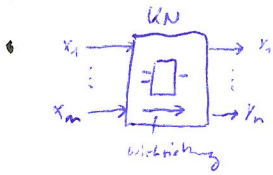
(iii) Festlegung AP:

$I_{B_0} = 0 \rightarrow$ AP0 (log. 1)

$I_{B_1} > 0 \rightarrow$ AP1 (log. 0)

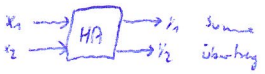
Kombinatorische Netzwerke:

Def.:
- KN realisiert log. Schaltfunktionen
- Eingangsvariablen bestimmen Ausgangsvariablen
- keine Speicherung! (→ keine Rückkopplung)



Beispiele:

(i) Halbaddierer:

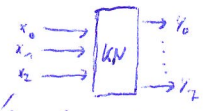


x_1	x_2	y_1	y_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$y_1 = \bar{x}_1 x_2 \vee x_1 \bar{x}_2$

$y_2 = x_1 x_2$

(ii) Codeumsetzer:



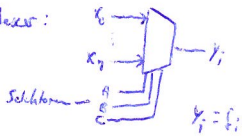
Binär codierte Ord.-Zahl (dez)

(1 aus 8) codierung

x_2	x_1	x_0	y_7	y_6	...	y_1	y_0
0	0	0	0	0	...	0	1
0	0	1	0	0	...	1	0
...							
1	1	1	1	0	...	0	0

3 bit - Binärdekodierer

(iii) a) Multiplexer:



$y_i = f_i(A,B,C) \cdot x_i$

b) Demultiplexer

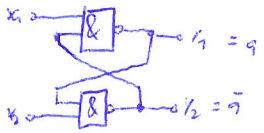
$y_i = f_i(A,B,C) \cdot X$

Flip Flops / Speicherbausteine:

Einzelspeicher f. 1 Bit:

- besitzt 2 stabile Zustände → Speicher f. 1 Bit Information
- besitzt 2 komplementäre Ausgänge

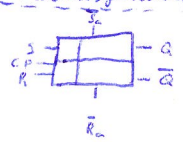
Flip Flop:

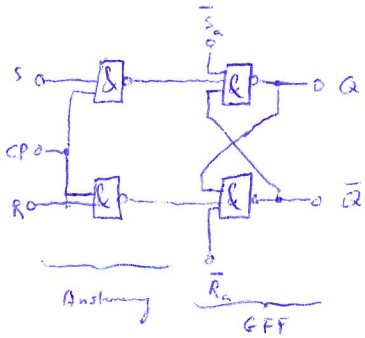


x_1	x_2	y_1	y_2	Bedeutung
0	0	1	1	keine, da $y_1 \neq \bar{y}_2$
0	1	1	0	Setzen d. Speichers
1	0	0	1	Reset d. Speichers
1	1	0(1)	1(0)	Speichern (keine Änderung)

Taktgesteuerte Speicher:

(i) Taktzustandsgesteuertes FF:

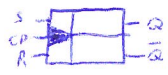


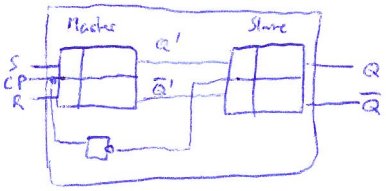


$\bar{S}_a, \bar{R}_a$: asynchron (taktunabh. Set-/Reset-eingänge)

S^n	R^n	Q^{n+1}
0	0	Q^n (Speichern)
0	1	0 (Reset)
1	0	1 (Set)
1	1	X (nicht def.)

(ii) Master-Slave-FF:





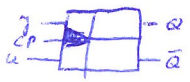
- 2-Speicher-FF
- geeignet für Verarbeitung
- bei posit. Taktflanke wird Info am Eingang eingelesen, bei neg. wird Info vorsigbert

(iii) D-FF (Delay-FF):

D^n	Q^{n+1}
0	0
1	1

$Q^{n+1} = D^n$

(iv) JK-FF:

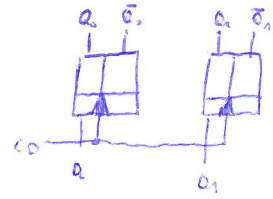


J^n	K^n	Q^{n+1}	
0	0	Q^n (Speichern)	SR-FF
0	1	0 (Reset)	SR-FF
1	0	1 (Set)	SR-FF
1	1	$\bar{Q}^n$ (Toggle)	

Charakt. Schalt-Fkt.:

$Q^{n+1} = (J\bar{Q} + \bar{K}Q)^n$

Sequenzielle Netzwerke: Prinzip eines Automaten: Bsp.: mod 4-Zähler:

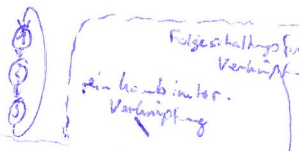


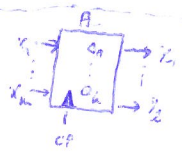
Realisierung: Sequ. Netzwerk; Abstrakt: Automat

Merkmale:
- Zustände d. Automaten gegeben durch Zustände seiner Speicherglieder
- Fortschalten in Abhängigkeit d. CP
- FKT darstellbar in Zustands-Übergangs-Diagramm:  → Übergänge

allg. noch abh. von Eingangswerten

- Bsp. f. Graph: (i)  (ii)



- **Mealy-Automat**: Def.: 

$\underline{X}$ ... Eingangswerte
$\underline{Y}$ ... Ausgangswerte
$\underline{Q}$ ... Zustandswerte

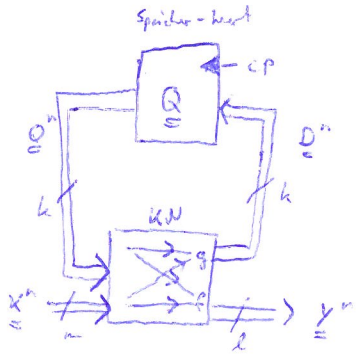
Ausgangsfkt.: $\underline{Y}^n = f(\underline{X}^n, \underline{Q}^n)$
Übergangs-Fkt.: $\underline{Q}^{n+1} = g(\underline{X}^n, \underline{Q}^n)$

Spezialfälle: $\underline{Y}^n = f(\underline{Q}^n)$ ... Moore-Automat (Ausgang zeitsynchron)
$\underline{Y}^n = \underline{Q}^n$ ... Medwedew-Automat

- **Realisierung**: Speicherglieder als k D-FF:

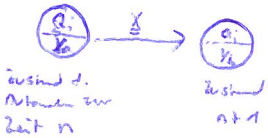
$\underline{Q}^{n+1} = \underline{D}^n$

damit: $\underline{Y}^n = f(\underline{X}^n, \underline{Q}^n)$ KN1
$\underline{D}^n = g(\underline{X}^n, \underline{Q}^n)$ KN2
$\underline{Q}^{n+1} = \underline{D}^n$ k D-FF

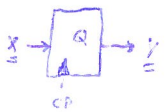


Sonderfälle: Moore-Automat: $\underline{Y}^n = f(\underline{Q}^n)$ ⇒ Änderung am Eingang $\underline{X}$ bewirkt keine direkte Änderungen am Ausgang $\underline{Y}$

Medwedew-Automat: $\underline{Y}^n = \underline{Q}^n$ ⇒ Ausgang d. Automaten entspricht Zustand

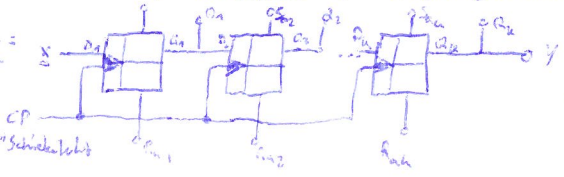
- **Moore-Graph**: 

- **Entwurfsprobleme**:
(i) Problemdefinition (Moore-Graph: k Zustände → z log k Speicherglieder)
(ii) Bestimmung d. Eingangsfunktionen $D_i = D_i(\underline{X}, \underline{Q})$ (Wahrheitstabelle)
(iii) Umsetzung in sequentielles Netzwerk

Klassen sequ. Netzwerke: (i) Zähl-/Steuerwerke: 

$\underline{X}$: Zählmodus, Auswahl d. Steuerprogramm
$\underline{Y} = \underline{Q}$ Zustand
$\underline{X}$ bleibt i. d. R. für die Dauer eines Ablaufs d. Steuerzyklus konstant

(ii) Hardware-Informationsverarbeitung: Rechenwerke: Multiplikationswerk
Divisionswerk
Generatoren: Prüfbitgenerator (Kanalcodierung)

(iii) Registerschieberketten: 

S_{0i}, R_{0i} ... Asynchr. Set-/Reset-eingänge für parallelen Einschreiben/Lesen
$Q_1 - Q_n$ ... Speicherinhalt f. parallelen Betrieb
$\underline{X}$ ... SR-Eingang, für seriellen Einschreiben } CP-
$\underline{Y}$ ... SR-Ausgang, für seriellen Auslesen } Takt

Rechenwerke: Einstellige Verknüpfungen: (i) Volladdierer: Übertrag c_i ← VA ← Operandstellen a_i, b_i; Summe s_i

a_i	b_i	c_{i-1}	c_i	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$\Rightarrow s_i = (a_i \equiv b_i) \not\equiv c_{i-1}$

$c_i = a_i b_i + (a_i + b_i)\, c_{i-1}$

(ii) Multiplikation: Mult (a_i, b_i) → Produkt p_i

a_i	b_i	p_i
0	0	0
0	1	0
1	0	0
1	1	1

$p_i = a_i b_i$

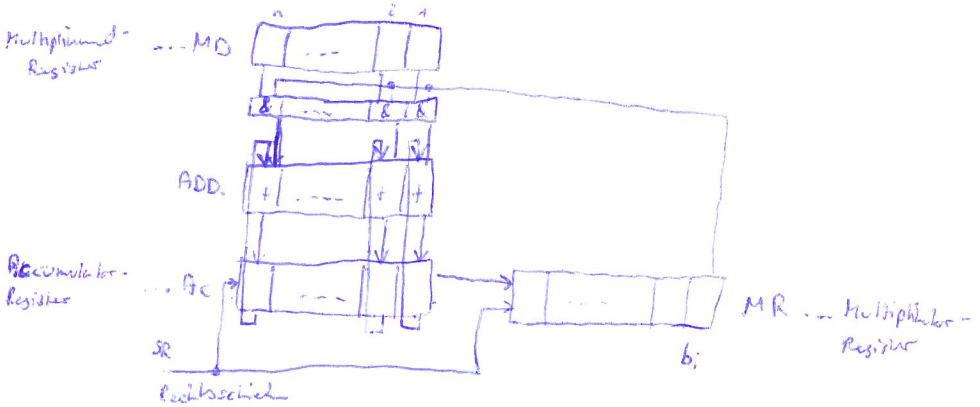
(iii) Volladdierer (Ripple carry): a_n, b_n … a_2, b_2, a_1, b_1 → VA … VA, VA; c_n, c_{n-1}, c_2, c_0; Ausgänge s_n, s_2, s_1

- sukzessive Übertragsverarbeitung
- Ergebnis: $2\tilde{c}$ ($\tilde{c}$ … Verknüpfungszeit)
- feine Kaskadierung
- Einschwingverhalten durch sukzessive Übertragsverarbeitung
- stabiles Ergebnis nach $n \cdot 2\tilde{c}$

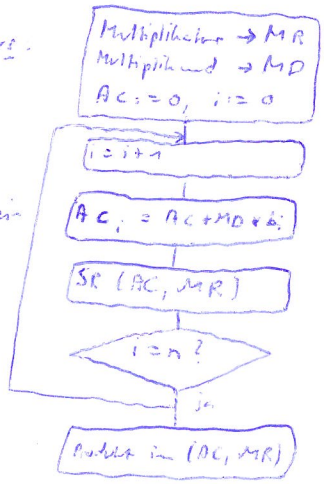
(iv) Subtraktion: Subtraktion = Addition d. Komplements:

2er-Komplement: Invertierung aller Stellen + 1

(v) Multiplikation: ≙ stellenweise Multipl. mit Stellen d. Multiplikators + stellenrichtige Addition auf d. Zwischenergebnis

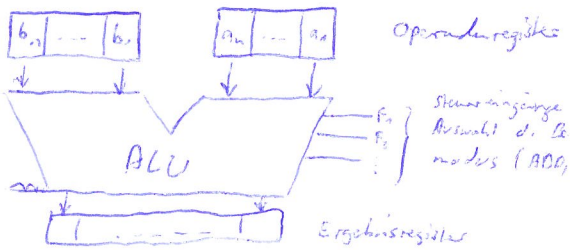


Algorithmus:



Beachte: Produkt i.a. doppelt so lang und wird durch Abschneiden der niederwertigen Stellen wieder auf Normallänge n gebracht (Fehler!)

(vi) Arithmetisch-logische Einheit (ALU):



ALU ist durch ein rein kombinatorisches Verfahren realisierbar

(vii) Periphere Massenspeicher: Zwei verschiedene physik. Prinzipien: • magnet. Remanenz - Induktion (Diskette, Platten-/Bandspeicher)
• Oberflächenveränderung d. Laserstrahl-Abtastung (CD, DVD)

Ein-/Ausgabe

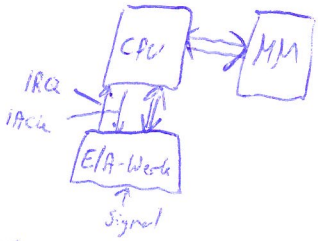
• Funktionen d. E/A-Werks: (i) Anpassung: • Geschwindigkeit zw. ZE und E/A-Gerät
• Datenformate, Blockgrößen
• Signalpegel, Codes
• Übertragungsprotokolle
• Prüfung
• Seriell-/Parallelwandlung

(ii) Initialisierung: Anstoß einer von außen angeregten E/A-Operation mittels Unterbrechung

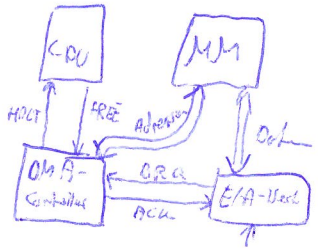
(iii) Steuerung: Selbstständige Abwicklung ~~eines vom Anfang bis zum Ende abgelaufen~~ E/A-Auftrags nach dessen Initialisierung parallel zur Arbeit d. CPU und Rückmeldung nach Abschluss d. Auftrags

• E/A durch R-/W-Operation auf die E/A-Einheit: E/A-Einheit wird über spezielle Befehle angesprochen direkt über d. laufende Anwenderprogramm. Dieses wartet bis zum Abschluss d. Vorganges.
⊕: Kein Synchronisationsproblem
⊖: CPU wenig effektiv genutzt, da E/A-Vorgänge i.a. sehr langsam sind.
(Eine spezielle Form dieser E/A spricht E/A-Einheit über den Adressbus an wie den Speicher ("memory mapped I/O")

• E/A durch Programmunterbrechung:
1. Interrupt Request:
• Unterbrechung d. Programmes
• Programmstatus in spezielles Register
• Analyse d. Unterbrechungsursache
2. Interrupt Acknowledge:
• Bestätigung d. Anforderung
3. Start d. Unterbrechungs-Behandlungsprogrammes
• Austausch v. Daten zw. ZE und E/A-Gerät unter Regie d. CPU / Unterprüfung-Behandlungsprogramm
4. Wiederaufnahme d. unterbr. Programmes
• Laden d. Status
• Fortsetzung d. Bearbeitung

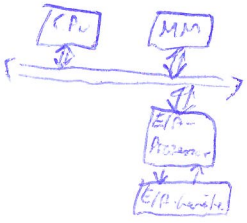


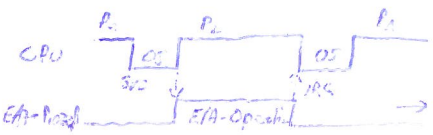
• E/A durch Direktspeicherzugriff (DMA):
1. DMA-Anforderungen
• Bus Request Signal BRQ
• Anhaltesignal an CPU HALT
• Bestätigung d. CPU FREE
• " d. DMA BACK
2. Datenübertragung:
• E/A-Werk greift über internen Bus direkt auf Speicher zu



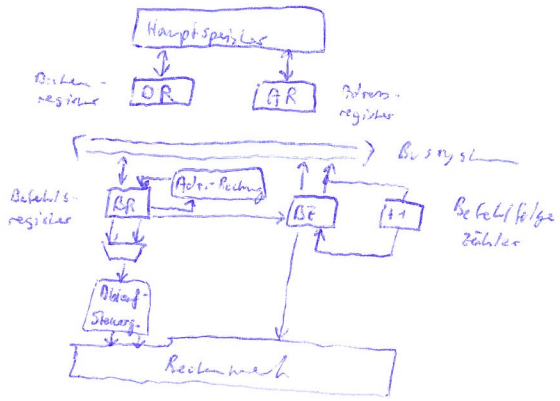
Arbeitsweise: • Transfer wortweise
• Übertragung ganzer Blöcke

• E/A durch spezielle E/A-Prozessoren
1. E/A-Anforderungen
• Unterbrechung d. Programmes (Supervisor Call SVC)
• Rettung d. Programmstatus
• Start eines E/A-Unterbrechungs-Behandlungs-Programmes auf das CPU
• Laden d. E/A-Prozessors mit Steuerdaten (Adressen, E/A-Geräteadressen etc.) und Anstoß d. E/A-Operation
2. Start eines anderen Benutzerprogrammes
3. Rückmeldung d. E/A-Prozessors nach abgeschlossener E/A
• Interrupt Request IRQ
• Unterbrechung d. laufenden Programmes und Rettung Prog.-Status
• Laden d. E/A-endlosen Programmes
• Fortsetzen d. ursprüngl. Programmes

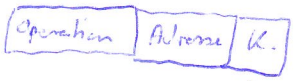




Steuerwerk: Das Befehlssteuerwerk dient d. Steuerung aller Einzeloperationen, welche für die Durchführung eines Maschinenbefehls erforderlich sind. → Maschinenbefehl zerfällt in eine Reihe sequentiell bzw. parallel ausführbarer Einzeloperationen (Mikro-Operationen)

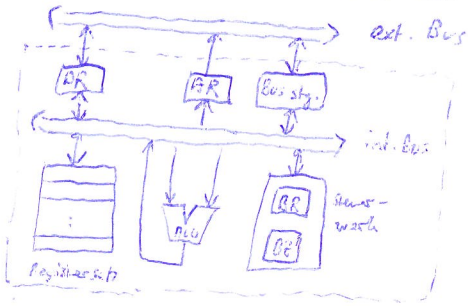


Befehlsaufbau:



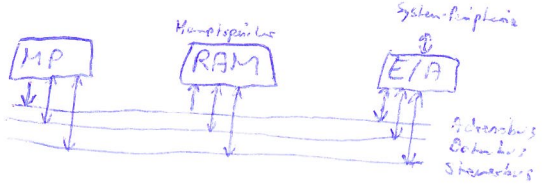
Kennungsteil für Adress-Interpretation

Mikroprozessor: besteht aus:
- Rechenwerk (Operandenverknüpfung)
- Steuerwerk (Befehlsverarbeitung)
- Register-Speicher (Zwischenspeicherung von Operanden / Befehlen)
- Schnittstelle (zum Speicher / E/A-Werken über Bus)



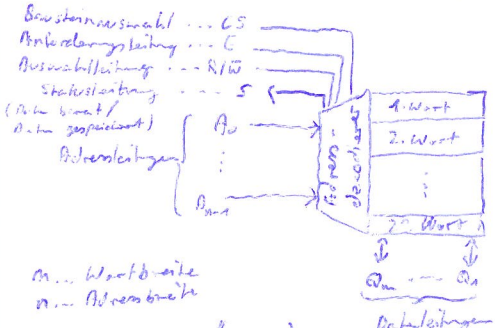
- hochintegrierter Chipbaustein
- Ablaufsteuerung Steuerwerk in Abhängigkeit eines Systemtakts durch den
- Programmierung in Assemblersprache

⇒ **Busbau zum Rechnersystem:**



- Hauptspeicher ist Schreib-/Lesespeicher zur Aufnahme v. Programmcode und Daten
- Bei Mikrorechner-Anwendungen wird das Programm häufig in einem Festspeicher (ROM) abgelegt
- Ein/Ausgabe möglich:
  - Zugriff auf Hauptspeicher/MP von außen durch Interrupt oder Direktspeicherzugriff (Direct Memory Access) zur Eingabe von Daten bzw. Ausgaben eines E/A-Vorgangs
  - Ausgabe von Daten auf peripheres Medium

Speicher: i) **RAM:** - Schreib-/Lesespeicher
- Information wurde bit-/bitgruppenweise (Wort, Byte) gespeichert



m ... Wortbreite
n ... Adressbreite

2^n Speicherkapazität (Worte)

$k = 2^{10}$: 1 024 ("Kilo")

$M = 2^{20}$: k·k ("Mega")

t_R = Zugriffszeit (Lesen)

T_W = Zykluszeit (Schreiben)

(ii) **Technologie:** Schaltkreistechnik: bipolar — TTL-Speicherzelle
unipolar — MOS-Speicherzelle

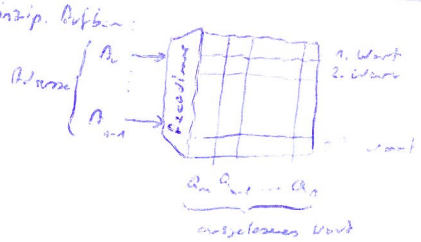
Speichertechnik: statisch — FF
dynamisch — Ladungsspeicher-prinzip mit Auffrischung

(iii) **Speichersteuerung:** a) Asynchrone Ansteuerung ("Handshake-Protokoll")

b) Synchrone Ansteuerung (komplexer Abschnitt einer Taktperiode)

Lese/Schreibvorgang innerhalb einer Taktperiode

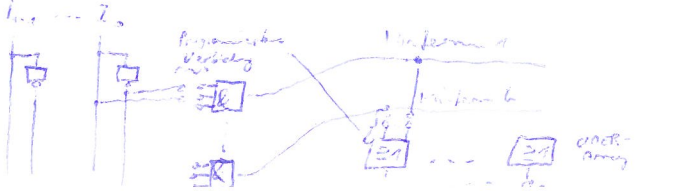
(iv) **Festwertspeicher:** Prinzip. Aufbau:



Arten: - ROM
PROM (programmierbar)
- Löschbar (Einzelprogrammierung)
Speicher

(v) **Programmierbare Logic Arrays (PLA):**
- Unterscheiden sich von Festwertspeichern durch höheren Programmieraufwand: PLA's sind aus einem UND- und einem ODER-Array aufgebaut, welche programmierbar sind
- Direkte Umsetzung von DNF's

$Q_n = a_{n1} (\bar{E}_1 \bar{E}_2 \dots \bar{E}_{n-1}) + a_{n2} (E_1 \bar{E}_2 \dots \bar{E}_{n-1}) + \dots + a_{nk} (E_1 E_2 \dots E_n)$ Min-Term



Objekte & Objektstrukturen: Objektbegriff: $\mathfrak{A} := (N, \mathfrak{I})$ mit $N \xrightarrow{\alpha} \mathfrak{I}$

N ... Nachricht / Bezeichnung d. Objekts
$\mathfrak{I}$ ... Information / Wert d. Objekts
α ... Interpretation
$\xrightarrow{\alpha}$ ... besitzt den Wert unter d. Interpretation α

- Objekte sind in gegebener Zusammenhang nicht weiter in Bestandteile zerlegbar
- Bsp: ganze Zahlen $\{\ldots -2, -1, 0, 1, 2, \ldots\}$
- Objektart: Zusammenfassung v. abzählbar vielen (unterscheidbaren) Objekten, auf denen gl. Operationen definiert sind, zu einer Klasse. Objektarten können auch als Objekte aufgefasst werden ($\mathbb{B}$, $\mathbb{Z}$, $\mathbb{N}$)
- Zusammengesetztes Objekt: n-Tupel $(O_1, \ldots, O_n)$ $n \geq 2$ von Objekten O_i d. Objektart N_i
- In Programmierung: Datentyp: Objektart
  Variable: Objekt, das d. Speicherung veränderlicher Objekte unter bestimmter Bezeichnung d. zugehörigen Variable enthält als Wert einen Bezug auf ein Objekt. Dieses Objekt ist über die Variablenbezeichnung zugänglich.
  Konstante: Objekt, das d. Speicherung eines festen Objekts unter einer bestimmten Bezeichnung dient
  Rechenvorschrift: Objekt, dessen Wert eine Vorschrift zur Lösung einer gegebenen Problemstellung repräsentiert

Operationen auf Objekte:
- Einstellige Operationen (z.B. Negation, Quadrieren, ...)
- Zweistellige Operationen (z.B. Addition, Subtraktion, ...)
- Prädikate (liefern als Ergebnis true/false)

Speziell zusammengesetzte Objekte (Objektstrukturen):
- Selektion = Zugriff auf einzelne Komponenten d. Objekte
- Selektoren = für diesen Zugriff erforderlichen Projektionsfunktionen

- Verbunde: zusammengesetzte Objekte, bei denen zusätzlich zu d. Komponentenwerten Bezeichnungen f. d. Selektoren definiert werden
  Schreibweise für Zugriff auf die durch Selektor s bestimmte Komponente eines Verbundes v: $v.s$
- Reihungen (Felder, Arrays): in Bez. zu Verbunden wird über Indizes zugegriffen
- Rekursive Objektstrukturen: Def. einer Menge von rekursiven, zusammengesetzten Objekten beinhaltet als Komponentenwert die zu beschreibende Objektart selbst

Mengen & Relationen: Begriffe: Potenzmenge: $\mathcal{P}(A)$ ist Menge aller Teilmengen von A (inkl. leere Menge und A selbst)

$$|\mathcal{P}(A)| = 2^n \quad (\text{wobei } n = |A|)$$

- Partition: Menge, deren Elemente disjunkte, nicht-leere Teilmengen von A sind, wobei jedes Element von A in genau einem Element d. Partition vorkommt.

Operationen: $A \cup B$ (Vereinigung)
$A \cap B$ (Durchschnitt)
$A - B$ (Differenz)

geordnetes Paar: 2-Tupel (a_i, b_j)
Kartesisches Produkt $C = A \times B$: Menge aller geordneten Paare d. Mengen $A \& B$

$(A \times B) \times C = A \times (B \times C)$
aber: $A \times B \neq B \times A$

Kardinalität: $|A|$ = Anzahl d. Elemente in A
$|A \times B| = |A| \cdot |B|$

Relationen: $R \subseteq A \times B$

Def.-Bereich: $Def(f)$ von R, wobei f das Tripel (X, Y, R) bedeutet.
$Def(f)$ ist def. als die Menge $\{x \in X\}$, für die mind. ein $y \in Y$ ex. mit geordnetem Paar $(x,y) \in R$

$$Def(f) : \{x \in X \mid \exists\, y \in Y \text{ mit } (x,y) \in R\}$$

Abbildung: $f(x)$ heißt Abb./Fkt. von x, wenn für ein $x \in Def(f)$ ein eindeutig bestimmtes $y \in Y$ ex. mit $(x,y) \in R$
(heißt total, wenn $Def(f) = X$)

Graphen

Bezeichnungen:

v_i … Knoten

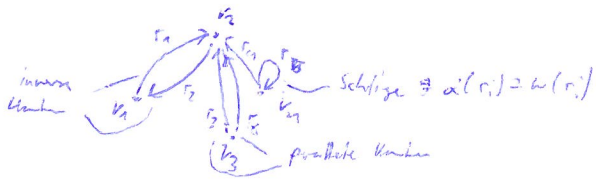
r_j … gerichtete Kanten

$V = \{v_1, \dots, v_n\}$ Knotenmenge

$R = \{r_1, \dots, r_m\}$ Kantenmenge

$\alpha(r_j)$ … Anfangsknoten von r_j

$\omega(r_j)$ … Endknoten von r_j



$G = (R, \alpha, \omega)$ (drei-Tupel aus Kanten, Knoten und Abbildungen α, ω von R in V)

Verbindungsmatrix: $Q = (q_{xy})$ mit q_{xy} = Anzahl aller r_i mit $\begin{cases} \alpha(r_i) = x \\ \omega(r_i) = y \end{cases}$

Vorgängermenge: $P(v_i)$

Nachfolgermenge: $S(v_i)$

Weg: $w = (r_{i_1}, \dots, r_{i_m})$ … endliche Folge von Kanten mit der Eigenschaft $\omega(r_{i_j}) = \alpha(r_{i_{j+1}})$ für $j = 1, 2, \dots, m-1$

$\alpha(w)$ … Anfangsknoten

$\omega(w)$ … Endknoten

$l(w)$ … Länge des Weges (= Anzahl der Kanten)

(elementarer Weg: kein Knoten doppelt)

Erreichbarkeit: v_j heißt erreichbar von v_i, wenn ein Weg w existiert mit $\alpha(w) = v_i$, $\omega(w) = v_j$

Zusammenhang: v_i und v_j sind stark zusammenhängend, wenn v_i von v_j und umgekehrt erreichbar sind.

Baum: Graphen mit minimaler Verbindung zwischen allen Knoten (= kein Kreis)

$|R| = |V| - 1$

Spezielle Bäume: Wurzelbaum: $G(V, R)$ ist Wurzelbaum wenn gilt:

i) $\exists v_0 \in V$ mit $P(v_0) = \emptyset$ (v_0 ist Wurzel)

ii) $\forall v_i \, (\neq v_0) \; \exists$ ein Weg von v_0 zu v_i

Blatt v_i: $S(v_i) = \emptyset$

Höhe: $h(v_i)$ = Länge des Weges von v_0 zu v_i

Söhne: $S(v_i)$

Vater: $P(v_i)$

Brüder: Alle Söhne mit gleichem Vater

Binärbaum: = Wurzelbaum, bei dem jeder Knoten ≤ 2 Söhne hat.

Es gilt: $k_G \leq 2^{h+1} - 1$ $\quad k_G = |V|$

$b_G \leq 2^h$ $\quad b_G$ = Blätteranzahl

$\hookrightarrow h \geq \ln(b_G)$

$h \geq \ln(k_G + 1) - 1$

Kombinatorik:

Permutationen: n Elemente $\Rightarrow P(n) = n!$ Permutationen

(α Elemente sind gleich $\Rightarrow P_\alpha(n) = \frac{n!}{\alpha!}$

Kombinationen: Anzahl von i aus n Elementen: $K_i(n) = \binom{n}{i} = \frac{n!}{i!(n-i)!}$

Variationen: $V_i(n) = \frac{n!}{(n-i)!}$ (bei Auswahl von i Elementen ex. $i!$ Permutationen)